TrustID: Trustworthy Identities for Untrusted Mobile Devices

Julian Horsch, Konstantin Böttinger, Michael Weiß, Sascha Wessel,
and Frederic Stumpf
Fraunhofer AISEC
Garching near Munich, Germany
{firstname.lastname}@aisec.fraunhofer.de

ABSTRACT

Identity theft has deep impacts in today's mobile ubiquitous environments. At the same time, digital identities are usually still protected by simple passwords or other insufficient security mechanisms. In this paper, we present the TrustID architecture and protocols to improve this situation. Our architecture utilizes a Secure Element (SE) to store multiple context-specific identities securely in a mobile device, e.g., a smartphone. We introduce protocols for securely deriving identities from a strong root identity into the SE inside the smartphone as well as for using the newly derived IDs. Both protocols do not require a trustworthy smartphone operating system or a Trusted Execution Environment. In order to achieve this, our concept includes a secure combined PIN entry mechanism for user authentication, which prevents attacks even on a malicious device. To show the feasibility of our approach, we implemented a prototype running on a Samsung Galaxy SIII smartphone utilizing a microSD card SE. The German identity card nPA is used as root identity to derive context-specific identities.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection

Keywords

Identity Derivation; Smartphone; Mobile Security; Combined PIN Entry; Secure Element; Identity Provider; Android; nPA

1. INTRODUCTION

In today's world characterized by personalized, Internet-driven services, every user possesses a multitude of digital identities for authentication to these services but also to physical entities. These identities are often presented in form of email addresses, user names or user identification numbers and secured by a diverse set of security mechanisms, e.g., passwords, Personal Identification Numbers (PINs), Transaction Authentication Numbers (TANs), and smart cards or other physical tokens in highly secure environments. The achieved security level of most of these mechanisms depends on how the user handles them. For example, the security of an identity protected by a password is only as strong as the password chosen by the user. Consequentially, identity theft is a widespread problem [8]. To address these issues, we propose the TrustID concept. Our four main contributions are:

1. An architecture for deriving and using context-specific IDs on mobile devices utilizing a Secure Element (SE) as credential store.

2. A protocol for secure ID *derivation* on a possibly unsecure or even malicious smartphone.

3. A protocol for secure ID *usage* on a possibly unsecure or even malicious smartphone covering several powerful use cases.

4. A secure combined PIN entry mechanism for user authentication *without* relying on a Trusted Execution Environment (TEE) in the smartphone or additional hardware like a trusted display/pinpad for the SE as opposed to, e.g., One-Time-Password (OTP) tokens.

In our concept, the user possesses *one* strong, long-term issued, generic root identity, called *RootID*. This RootID could, for instance, be a government-issued Electronic Identity Card (EIC). The user then uses this RootID to derive all kinds of different, context-specific IDs storing them in a certified SE in his off-the-shelf smartphone. The TrustID ID derivation protocol relies on a trusted third party, the Trusted Identity Provider (TIP), to decide about ID requests based on the information stored in the RootID. The derived IDs can then be used via all interfaces the smartphone provides, typically including Near Field Communication (NFC) but also the Internet, while the RootID remains at the user's home or some other secure place avoiding the risk of loss.

Moving the security basis away from the user towards highly secure SEs substantially increases overall security. Most users already own a smartphone and carry it with them, so its the most natural place to store the IDs the user needs on a daily basis. The certified SE can be included into the smartphone flexibly in different form factors, including

embedded SEs, microSD cards and Universal Integrated Circuit Cards (UICCs). The concept also takes into consideration that there may be several different SE manufacturers who are allowed to produce SEs certified for the TrustID system. Nevertheless, using the user's smartphone, which is an untrusted and possibly malicious device, in a security relevant context requires some further security measures to be taken into consideration. We therefore propose a *secure combined PIN entry* mechanism, which allows to establish a secure channel between SE and RootID by using the corresponding secret PINs without the user entering them in clear via the smartphone's untrusted touchscreen. So the security of the ID derivation mechanism and the resulting ID does *not* rely on the security of the smartphone software stack, which is increasingly important since the past years show the growing relevance of smartphones as targets for diverse attacks [6, 1]. If the user loses his smartphone, he can use his RootID to revoke the IDs stored in the SE of the lost device. Our concept covers several use cases such as financial transactions, physical access control, car-sharing access to access control for online services.

This paper is organized as follows. In Section 2, we describe our attacker model. In Section 3, we give an overview of existing research results. Section 4 introduces the basic concept, architecture and the protocol for the TrustID ID derivation and usage. Aspects of our prototype implementation are discussed in Section 5. Section 6 evaluates our protocol regarding its security. We conclude in Section 7.

2. ATTACKER MODEL

The attacker has complete control over the entire network. On the smartphone, he is able to modify software, to run arbitrary programs, to read out RAM, and to eavesdrop bus systems. Attacks on the SE as well as the RootID are out-of-scope because these devices are hardened against software and hardware attacks. The attacker has no knowledge about the PINs protecting the SE and the RootID, but he might be able to gain knowledge of the combined PIN. Attacks on the TIP are also out-of-scope.

3. RELATED WORK

Leicher et al. [12] provide an OpenID based approach for mobile SSO across different devices. In contrast to our approach, their concept relies on the Mobile Network Operator (MNO) as root for the derivation of mobile identities and on a fully trusted mobile device.

Urien et al. [13] provide a mobile ID in form of an EAP-TLS smart card. Similar to our approach they store a key pair as identity credential in the smart card. For securing the communication, they use the TLS stack of those special smart cards to form secure channels between the different entities of their protocol. However, they do not scope a compromised smartphone in any case.

In [11], Hyppönen describes a protocol for deriving an ID from an identity issuer which is sending the credentials to the SE over an identity proxy (a mobile phone). Nevertheless, this approach does not deal with identity theft by the means of a relay attack as described in our security evaluation (see Section 6) or with a possible compromisation of the mobile device. Chen et al. [5] propose a mobile payment ID derived from the Citizen Digital Card (CDC), which is a governmental PKI-based ID card, onto an NFC smartphone.

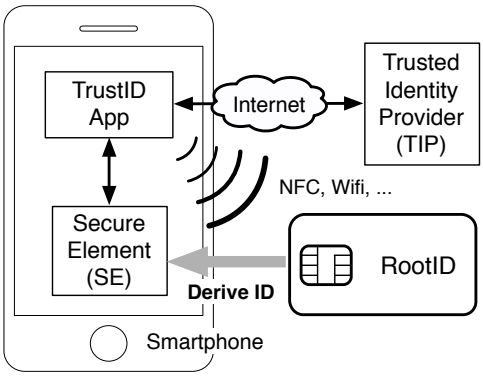

Figure 1: Overview of the TrustID system

This is similar to our scenario of ID derivation. However, their protocol relies on the MNO as trusted ID provider as he owns the SE used in their scenario (the SIM card). Further, the PIN for the CDC is entered in clear allowing an attacker compromising the smartphone to eavesdrop it.

Dmitrienko et al. [7] provide an approach for delegable access control utilizing NFC-enabled smartphones. Their system is not based on a RootID, but relies on users directly delegating their credentials to each other. In contrast to our approach where the smartphone can be completely untrusted, they rely on a large trusted computing base.

Summarizing, none of the previous contributions covers the full process of deriving multiple identities from a strong root identity employing a trusted identity provider, that might be integrated in a governmental PKI, while taking a compromised smartphone into account.

4. CONCEPT

The goal of the proposed TrustID architecture and protocol is to derive a context-specific identity from a RootID by interacting with a remote Trusted Identity Provider (TIP) and to store the derived ID into a SE inside a smartphone. An app on the smartphone acts as a hub and mediates the communication between the different entities without being able to discover any sensitive information. An *identity* in our concept basically consists of an asymmetric key pair and an associated certificate containing all additional information about the context and usage of the identity. The entities which are involved in the derivation process and their connections are depicted in the system overview in Figure 1. These entities are described in the following:

Smartphone. The smartphone mainly acts as a carrier for the SE and as runtime environment for the TrustID app providing the hardware interfaces necessary for the communication with the other entities (e.g., Wifi, NFC, SDIO) and with the user (via the touchscreen).

Secure Element. The SE (e.g., a microSD card or UICC) used in the TrustID protocol must be *certified*, i.e., must contain some key material for which the TIP is able to do a verification. This is discussed in detail in Section 4.3. This key material is *not* user-specific, but is used to provide a guarantee that the newly derived

ID is stored in an actual SE which matches the required security standards. Therefore, such SEs could be distributed/sold to the users without any personalization or pre-provisioning.

TrustID App. The TrustID app on the smartphone acts as a hub for all communication between the other entities and is responsible for presenting information to and getting input from the user via its User Interface (UI). In the TrustID protocols, all software on the smartphone can be untrusted because of the special PIN entry mechanism described in Section 4.3.

RootID. The RootID is some kind of long-term issued smart card, which securely stores the personal data of the user. The RootID must contain key material to authenticate itself towards the entity reading the data and to make sure that only authorized entities may read out its data. A typical example for RootIDs fulfilling these requirements are government-issued EICs like the German identity card *nPA* used in our prototype implementation (see Section 5).

Trusted Identity Provider. The TIP provides the backend of the system architecture and is connected to the smartphone through the Internet. In the ID derivation protocol, the TIP must decide if an identity request by a user is fulfilled based on the user's RootID and the SE in the user's smartphone. The TIP is also responsible for generating the certificate for the new ID.

4.1 Architecture

During ID derivation, several logical channels are established. Figure 2 shows the system architecture with named channels between the entities. We differentiate channels which conceptually directly connect two entities (solid lines) and channels which are established between two entities through a third TrustID entity (dashed lines):

RootID-Channel. This channel connects the TrustID app with the RootID. It is *unprotected* and is realized differently depending on the smartphone hardware and RootID form factor. For example, the channel could be established using an NFC connection to a contactless smart card RootID.

SE-Channel. This channel connects the TrustID app with the SE. It is *unprotected* and established differently depending on the SE form factor, e.g., via Secure Digital Input Output (SDIO) in case of a microSD card.

TIP-Channel. This channel connects the TrustID app with the TIP. The channel is established via the Internet and does *not* have to be protected, but can optionally use Transport Layer Security (TLS).

TIP-SE-Channel. This channel connects the TIP and the smartphone's SE. It is mutually authenticated and encrypted and established based on static asymmetric keys present in the involved entities.

RootID-SE-Channel. This channel connects the RootID and the smartphone's SE. Similar to the TIP-SE-Channel above, this channel is also mutually authenticated and encrypted but uses the PIN mechanism described in Section 4.3 to be established.

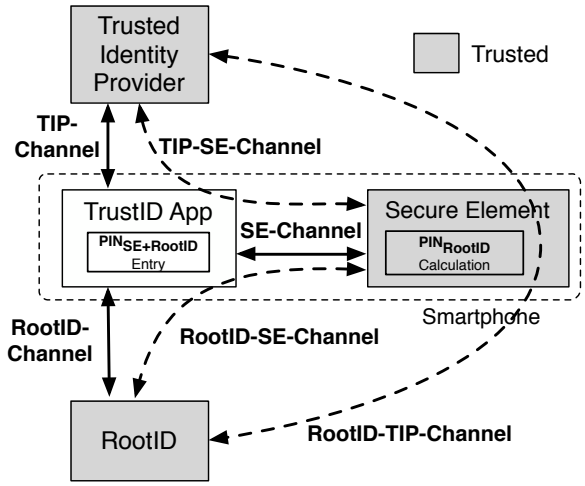

Figure 2: TrustID system architecture

RootID-TIP-Channel. This channel connects the RootID and the TIP. The channel is mutually authenticated and encrypted. It is established through the TIP-SE-Channel and the RootID-SE-Channel using the static key material in RootID and TIP. After this channel is established, any further communication between TIP and RootID is routed through this channel, which therefore replaces the two previous logical channels.

Attacks on these channels are discussed in the security evaluation in Section 6 after introducing the ID derivation protocol in detail in the next sections.

4.2 Protocol Prerequisites

As mentioned before, there are some prerequisites regarding the key material which has to be present on the entities to successfully run the TrustID protocol for ID derivation.

First, the **RootID** must contain an asymmetric key pair (e.g., ECC or RSA) SK_{RootID} (private), PK_{RootID} (public). For the public key PK_{RootID} the RootID additionally contains a certificate issued by the TIP or some other Certificate Authority (CA) which can be validated by the TIP depending on the used Public Key Infrastructure (PKI). Furthermore, the RootID contains the public key of the root of the PKI. In our concept description, the TIP is assumed to be the issuer of the certificate $Cert_{TIP}(PK_{RootID})$ as well as the root of the PKI to simplify the protocol discussion. Access to the RootID is protected by a PIN called PIN_{RootID}.

Second, the **SE** must contain key material quite similar to the one in the RootID, i.e., an asymmetric key pair (SK_{SE}, PK_{SE}), a certificate for the public key and the public key of the CA in charge. The certificate is issued by the manufacturer of the SE and vouches for a genuine and authentic SE. The SE contains a PIN called PIN_{SE} to be used in the ID derivation protocol and depending on the chosen usage protocol (see Section 4.4) an additional PIN_{Use} protecting usage of the derived IDs. Again, for simplicity, we use the TIP as CA but the PKI can be designed quite flexible. The only requirements regarding the overall PKI are:

- The TIP must be able to validate the certificates of both, the RootID and the SE.

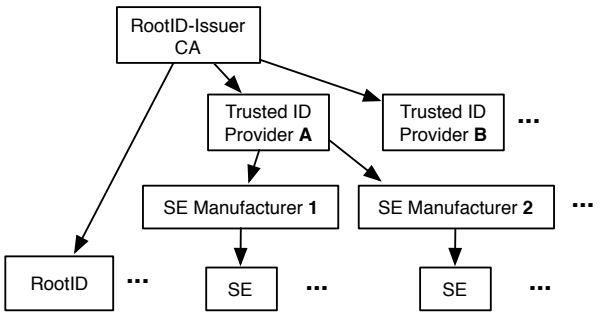

Figure 3: An example PKI for the TrustID system

- The TIP must be able to provide a certificate or a certificate chain which can be validated with the root CA keys contained in the SE and RootID (in our protocol description PK_{TIP} for both).

A simple example for a PKI is depicted in Figure 3. The depicted PKI allows for a system in which there are different TIPs, which are certified by the RootID issuer. The TIPs themselves certify different SE manufacturers.

4.3 Protocol

The real-life scenario for the TrustID protocol for ID derivation is the following: A user wants to store a context-specific identity in his smartphone, for example, some kind of virtual bank card, to get access to this specific service with his smartphone and without having to carry around his RootID and/or physical bank card. Therefore, the user starts the TrustID app on his smartphone, which he previously equipped with some certified SE (see Section 4.2), and generates the specific bank card request via the UI. This is where the TrustID ID derivation protocol starts. The protocol is depicted in Figure 4 and its single steps are described in the following. The key material prerequisites described in the previous section are also summarized in Figure 4 below each entity. At the end of the protocol, the new identity is stored securely in the certified SE inside the smartphone.

4.3.1 TIP-SE-Channel Establishment

In a first step, the SE and TIP establish the mutually authenticated and encrypted TIP-SE-Channel. The TIP-SE-Channel is established by using the static keys of both entities. With these, many different channel establishment protocols are possible, for example, a mutually authenticated certificate-based form of TLS. Since the SE is completely passive, all communication must be relayed by the smartphone and converted between the different transmission formats, for example, from Application Protocol Data Units (APDUs) to Hypertext Transfer Protocol (HTTP) requests and vice versa.

4.3.2 Secure Combined PIN Entry

A special PIN mechanism is used to establish the RootID-SE-Channel between RootID and SE. As described in Section 4.2, both the RootID and the SE each have a secret PIN (PIN_{RootID} and PIN_{SE}). These PINs are known to the user, who receives them together with the RootID/SE. Instead of entering one of the PINs directly via the touchscreen and therefore exposing it to the possibly malicious software stack of the smartphone, the user has to do a simple calculation on both PINs to derive a *combined* PIN named $PIN_{SE+RootID}$. This calculation is denoted in Figure 4 as function f and takes PIN_{SE} and PIN_{RootID} as arguments:

$$PIN_{SE+RootID} = f(PIN_{SE}, PIN_{RootID})$$

The function should be chosen to be easily calculable by the user. The calculation must be invertible by the SE with the knowledge of PIN_{SE} in the next step. The function is therefore formally parameterized by PIN_{SE} resulting in the function $f_{PIN_{SE}}$ which only takes PIN_{RootID} as single argument. The SE receives the combined $PIN_{SE+RootID}$ and because of its knowledge about PIN_{SE} is able to calculate PIN_{RootID} with the inverse function $f^{-1}_{PIN_{SE}}$. To put it another way, only the SE intended to store the ID by the user is able to calculate the secret necessary to establish a channel to the RootID. After this calculation, both entities share a secret password in form of PIN_{RootID}, which never went through the possibly malicious software stack on the smartphone. This password is then used in the next step to establish the RootID-SE-Channel. The combined PIN entry binds the SE and RootID to each other for the protocol run preventing certain relay attacks as discussed in the evaluation in Section 6.

To make the PIN mechanism more clear, consider the following simple but realistic example for the PIN calculation function f:

$$f(x, y) := x + y$$
$$f_{PIN_{SE}}(x) := PIN_{SE} + x$$
$$f^{-1}_{PIN_{SE}}(y) := y - PIN_{SE}$$

In a concrete example with this specific PIN function, we assume $PIN_{SE} = 123456$ and $PIN_{RootID} = 345678$. The user uses the introduced function and calculates

$$PIN_{SE+RootID} = f_{PIN_{SE}}(345678) = 469134$$

and enters it through the untrusted touchscreen. The SE then calculates

$$PIN_{RootID} = f^{-1}_{PIN_{SE}}(469134) = 345678$$

to obtain the shared secret for the channel establishment with the RootID. This example also shows that the calculation to be done by the user does *not* have to be complicated.

Both, SE and RootID, must implement try counters for the PIN entry to prevent brute force attacks.

4.3.3 RootID-SE-Channel Establishment

After the secure combined PIN entry, the SE and RootID share a secret password PIN_{RootID} which they use in the next step to establish the secure RootID-SE-Channel. For that, a balanced Password-Authenticated Key Agreement (PAKE) protocol is used [3, 4, 10]. Such a protocol allows the two parties to derive a common cryptographic key based on a common human-memorable password. In our aforementioned example and implementation scenario where the RootID is the German ID card nPA, the PAKE protocol used is the Password Authenticated Connection Establishment (PACE) protocol [9], which is the standard way to establish a PIN based connection to the nPA. In this context, the SE in a way takes the role of an nPA terminal.

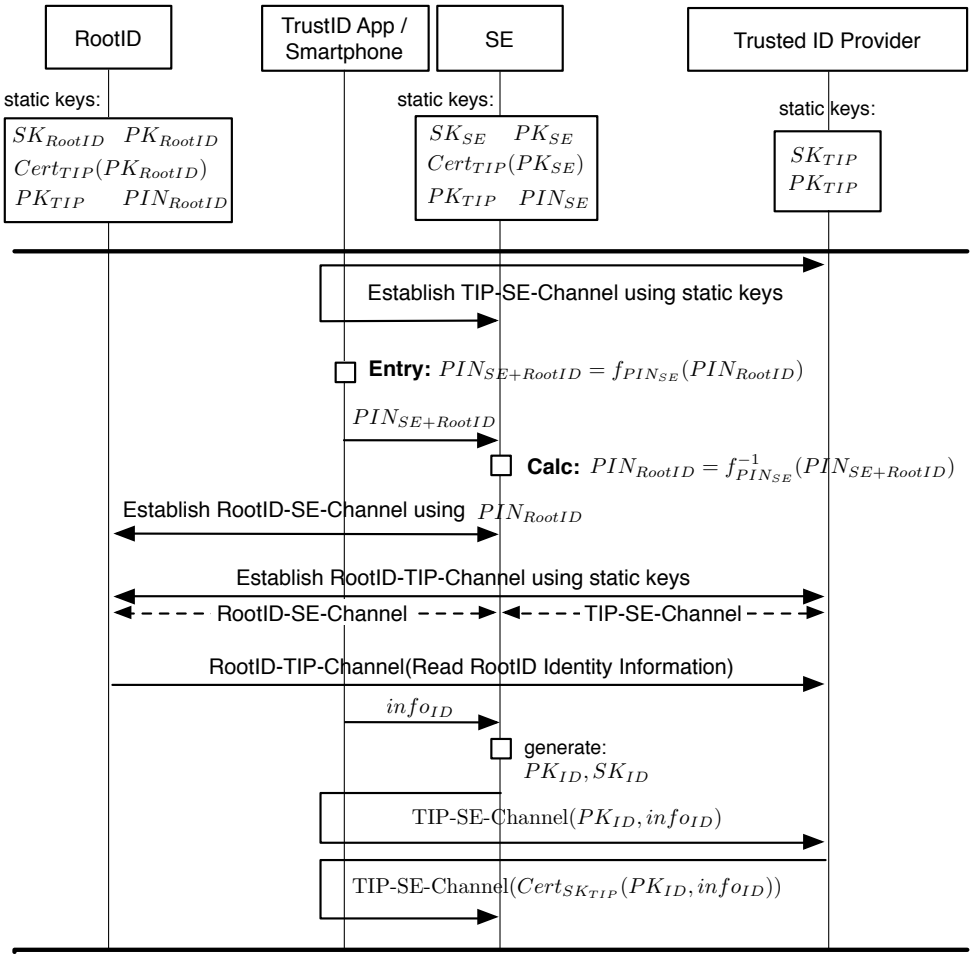

Figure 4: Protocol for the identity derivation

4.3.4 Read-out of the RootID Information

After the establishment of the RootID-SE-Channel and the TIP-SE-Channel, the RootID can be reached by the TIP going over the SE without data being sent unencrypted or unauthenticated at any point. The SE acts as a channel endpoint for both channels and tunnels the communication accordingly. This tunnel is used for establishing another channel, the RootID-TIP-Channel which allows an end-to-end encrypted and mutually authenticated communication between RootID and TIP, i.e., without even the SE being able to eavesdrop information. Just as for the TIP-SE-Channel, the available key material in RootID and TIP allow for a certificate-based channel establishment protocol similar to TLS or the like. The newly set up RootID-TIP-Channel then replaces the two channels ending in the SE for all communication between RootID and TIP and is used to read the user's identity information stored in the RootID.

4.3.5 ID Generation, Validation and Transmission

As soon as the TIP has successfully read the RootID identity information, the derived ID generation is started. Therefore, the TrustID app on the smartphone encapsulates information regarding the identity requested by the user (via the UI) into an $info_{ID}$ data structure and sends it to the SE. The SE stores the $info_{ID}$ object and generates an new key pair (PK_{ID}, SK_{ID}) for the new ID. It is important that the key pair is generated directly on the SE to make sure that the private key for each ID does not *ever* leave the SE. Afterwards, the public key PK_{ID} is sent together with the $info_{ID}$ object encrypted and authenticated through the previously established TIP-SE-Channel to the TIP. The channel ensures that the TIP *only* accepts a public key for certification which was generated in the particular SE with which the protocol was previously initiated.

The TIP is now responsible for validating the ID request based on the information read from the RootID and the information to be contained in the new derived ID, i.e., the $info_{ID}$ object. The actual mechanism for the decision about granting or refusing an ID request is highly context-dependent and therefore not specified in the presented protocol. The TIP could, for example, consult a backend at a bank before granting a virtual bank card ID to a user.

If the TIP grants the request, it generates a certificate containing the public key for the newly derived ID as well as the $info_{ID}$ object. So the ID is always bound to the specific context it was created for. The certificate is then sent via the TIP-SE-Channel to the SE where it is stored together with the ID's key pair.

4.4 ID Usage

With the asymmetric key pair and the certificate for the public key including context-specific information ($info_{ID}$), the newly derived ID provides everything to be used in diverse, and application-specific ways. Basically, there are two major cases to be differentiated for a terminal using the derived ID: Either only the ID authenticates itself towards the terminal or the authentication is done mutually. In the first case, it is sufficient for the terminal to have access to the certificate chain by which the ID was certified to be able to validate the ID. In the second case, the terminal infrastructure must be included into the PKI (see Figure 3), so that every terminal gets an own certificate which can be validated by the SE using its pre-installed public root key, e.g., PK_{TIP} in our concept.

The IDs stored on the SE must be protected from unauthorized usage. Under our general assumption that the smartphone might be compromised, this protection can be provided using different PIN-based schemes. Note that it might also be possible to have no PIN protection for less security-critical identities and use cases where it is sufficient to have the smartphone as token without proving any knowledge (e.g., cafeteria micro-payments). We propose two different PIN-based schemes.

4.4.1 Basic Protection Scheme

In the first scheme, the usage of the IDs stored on the SE is protected by a *single* PIN, which we call PIN_{Use} in the following. This PIN is distinct from the PIN_{SE} used for the ID derivation. When an ID is to be used by a terminal, the user approves the usage by entering this PIN_{Use} on the *terminal* which is assumed to be trustworthy. The PIN_{Use} can then be used as shared secret for a PAKE protocol between SE and terminal like described previously for the ID derivation protocol. The untrusted smartphone is not able to gain knowledge about the PIN_{Use}.

4.4.2 Reverse Secure Combined PIN Entry

The first scheme requires the terminal to be fully trusted because it gains knowledge of the single PIN_{Use} necessary for unlimited use of all IDs stored in the SE. The second scheme allows to restrict this trust in the terminal by using a mechanism which we call *reverse secure combined PIN entry*. This mechanism is quite similar to the PIN entry mechanism described in the course of the ID derivation but reverses it. The steps are the following:

1. The user initiates the usage of an ID, e.g., by choosing the ID and putting the smartphone on a terminal reader. The terminal and/or the smartphone app send a usage request for the ID to the SE.
2. The SE generates a random *one-time valid* PIN_{ID}, calculates the previously introduced (see Section 4.3.2) function $f^{-1}_{PIN_{SE}}$ on this PIN to produce a combined PIN_{SE+ID} and returns it.
3. The combined PIN is presented to the user on the display of the smartphone. The user is able to reverse the calculation with the knowledge of the secret PIN_{SE} using $f_{PIN_{SE}}$.
4. The user enters the calculated PIN_{ID} on the terminal which is then able to use it in a PAKE protocol to establish a channel to the *specific* ID on the SE.

The first approach is quite easily usable requiring the user only to memorize an additional PIN_{Use} but requires the terminal to be more trustworthy than in the second approach. The second approach provides the improvement that the terminal only gains knowledge of a one-time valid shared secret. It furthermore restricts the access to only one specific ID. Nevertheless, the user *cannot* be sure that the used ID is the one he expects without any trust in either the terminal display or the smartphone display. As a partial solution to this problem, it could be required that both displays have to show which ID is about to be accessed. Furthermore, it must be ensured that an attacker might not gain access to the PIN_{SE+ID} displayed on the smartphone and the PIN_{ID} entered on the terminal *at the same time*. This would allow him to calculate the secret PIN_{SE} only known to the user and the SE. Both attacks require control over the terminal and the smartphone at the same time, significantly reducing their potential.

5. IMPLEMENTATION

Our prototype implementation uses the government-issued German ID card nPA as RootID, an Android smartphone running the TrustID App and a JavaCard-based SE in a microSD form factor in the corresponding slot of the smartphone. Due to the architecture necessary for the usage of the data stored in the nPA via the eID protocol [9], the TIP in the implementation consists of two distinct entities: The eID-Server and the actual TIP. In the following we discuss important aspects of the implementation regarding the different involved entities.

5.1 Trusted Identity Provider

As mentioned before, the TIP from the concept is divided in two distinct entities in the implementation. There is the actual TIP which is under our control and there is the eID-Server which is responsible for reading-out the nPA on behalf of our TIP. All communication is conducted via HTTP(s) requests/responses, which the app on the smartphone translates into APDUs to be sent to the SE or RootID and vice versa. The TLS protection on this layer is optional as the security of the communication is enforced by the TrustID protocol on top of it.

5.2 Applet on the SE

The applet in the prototype is running on a G&D Mobile Security Card SE 1.0. This SE provides the JavaCard API in version 2.2.1 and the applet is therefore programmed against this API version.

Main task of the applet is to store IDs and to provide functions to derive, manage and use them. An ID in the applet basically consists of an asymmetric RSA key pair (currently 2048 bit) and a certificate (X.509) issued by the TIP for the public key, which additionally contains information about the ID ($info_{ID}$) as X.509 extensions.

As a basic restriction of the current version of the prototype, the RootID-SE-Channel does not terminate in the JavaCard applet but directly in the TrustID Android app. This is mainly due to the fact that the JavaCard API of currently available microSD card SEs does *not* support the specific Elliptic Curve Cryptography (ECC) operations necessary for the PACE protocol channel establishment, which is the PAKE protocol to access the nPA. With an SE available supporting the required functionality, it would be nec-

essary to port the related parts of the Android app into the SE applet to provide a complete implementation of the concept. Since the other parts of the prototype can be reused, this should only involve a moderate effort.

5.3 Android App

In the center of the implementation is the TrustID Android app. In the current prototype, the app is running on a Samsung Galaxy SIII (i9300) device with Cyanogenmod. The app provides a Graphical User Interface (GUI) which shows a list of IDs stored on the SE. The GUI allows to delete and activate the present IDs and to derive new ones. The usage of IDs is realized via the Card Emulation feature provided by the Cyanogenmod 10 release for the Galaxy SIII. This feature allows the smartphone to act as a contactless, i.e., NFC enabled, smart card while forwarding all incoming communication to software and vice versa. The TrustID app forwards this communication to the SE. This means that using one of the IDs stored on the SE is as easy as activating it and placing the smartphone on a reader.

The establishment of the secure channel between eID-Server and nPA is realized in the app using the open source library eIDClientCore [2].

Because of incompatibilities between the NFC hardware of the nPA and the Galaxy SIII, the prototype uses a so called *relay host* to enable their connection. An off-the-shelf smart card reader is connected to this relay host which reads the nPA and relays communication to the smartphone via wireless LAN.

The app listens for events indicating an NFC reader in reach and as soon as such an event is triggered provides a card emulation relay from this reader to the SE inside the smartphone. Therefore, it is only necessary to implement the ID usage functionality in the SE applet from where it can be directly used by a terminal reading the smartphone via card emulation.

6. SECURITY EVALUATION

In the following, we analyze the different channels and entities in the architecture for ID derivation (see Figure 2) regarding potential attacks and how the protocol prevents them. Security aspects for each **entity** are examined in the following based on our attacker model (see Section 2):

Trusted Identity Provider. This entity is assumed to be trusted and uncompromised and attacks against the backend are out of scope of the evaluation.

Secure Element and RootID. These entities are assumed to be trusted and unmodified. This is reasonable since they are distinct pieces of hardware hardened against software as well as physical attacks. Furthermore, both are protected by a PIN including a try counter.

TrustID App. This part of the system is assumed to be untrusted, i.e., may be maliciously modified. This must especially be considered in the evaluation with regards to user input/output.

We differentiate conceptually *direct* connection channels, i.e., the SE-Channel, the TIP-Channel and the RootID-Channel (solid lines in Figure 2), and the *indirect* connection channels, i.e., the TIP-SE-Channel, the RootID-SE-Channel and the RootID-TIP-Channel (dashed lines). Their security is evaluated in the following:

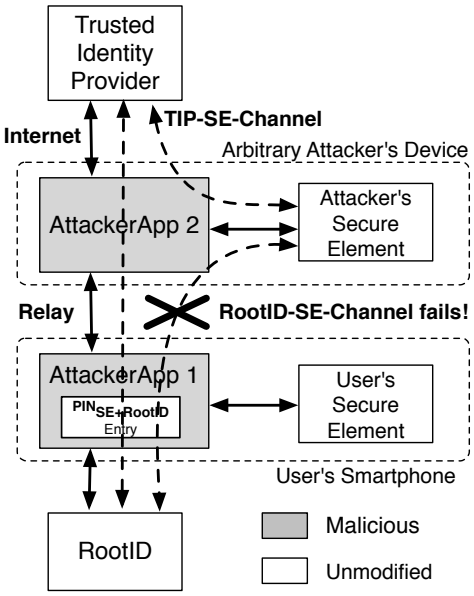

Figure 5: Relay attack and its prevention

Direct Channels. All direct channels are unprotected, i.e., provide neither authentication nor encryption or integrity protection. They can more or less be thought of as physical connections and attacks are prevented by the secure channels on top of them.

TIP-SE-Channel. This channel is established using the static key material in a TLS-similar protocol. It is therefore protected against eavesdropping, manipulation and replay attacks. Since it is a remote channel via the Internet, the channel is implicitly relayed without introducing a security problem. With the authentication of the SE, the TIP can be sure that the channel ends in a *certified* SE but at this point there is *no* association to a specific user. This leads to the relay attack depicted in Figure 5. There, the attacker modifies the app on the user's smartphone to relay the communication remotely to an attacker's device also containing a certified SE. The goal of the attack is to have the user authenticate an ID derivation (with RootID and secret PINs) but storing the new ID in the attacker's SE. Since the TIP-SE-Channel is only guaranteed to be established with *any* certified SE, the attacker is able to set it up to his own SE. Still, the attack is prevented with the RootID-SE-Channel and the secure combined PIN entry as described in the following.

RootID-SE-Channel. The secure combined PIN entry mechanism (see Section 4.3.2) is used to establish the RootID-SE-Channel. This mechanism makes sure that the shared secret in form of the PIN_{RootID} can only be gained by the SE for which the user calculated the combined $PIN_{SE+RootID}$. This effectively binds a specific SE and RootID as intended by the user to each other for one protocol run. Since the attacker's SE, despite its knowledge of $PIN_{SE+RootID}$, is *not* able

to calculate the shared secret, it cannot establish the RootID-SE-Channel. In other words, the RootID will not accept any connection attempt from the attacker's SE, effectively preventing the relay attack in Figure 5. By entering the combined PIN, the user furthermore guarantees the physical proximity of the three entities smartphone, SE and RootID, preventing also other relay attacks between these. Eavesdropping, replay and manipulation are prevented by using a secure PAKE protocol to establish the channel.

RootID-TIP-Channel. This channel is set up *through* the other secure channels using a TLS-similar, mutually authenticated certificate-based protocol. By establishing the channel through the RootID-SE-Channel and TIP-SE-Channel the different protocol steps are bound together, effectively preventing interleaving attacks.

The attacker might be able to modify the ID request generated by the user which would result in a different derived ID than intended by the user. Even if the attack is not detected by the TIP when validating the request against the RootID information, the wrongly derived ID would still *not* be under control of the attacker and safely stored in the user's SE. The attacker might be able to continously generate valid IDs on a compromised device containing a valid SE as soon as he gains knowledge of the $PIN_{SE+RootID}$. However, this attack is only possible as long as the RootID is in reach and, again, can be considered uncritical as the attacker does not gain control over the derived IDs. The attacker might also manipulate the displaying of IDs present on the SE, which we regard to be uncritical with the same argument as for the previous attacks. The same holds for all kinds of manipulations of communication data going through the app, including the $PIN_{SE+RootID}$.

7. CONCLUSION

In this paper, we introduced the TrustID architecture and protocol. Our approach allows for secure storage, derivation and usage of multiple context-specific identities on a possibly insecure or even malicious mobile device utilizing a SE as credential store. In the core of our concept, we introduced the secure combined PIN entry mechanism for user authentication, which does not rely on a trustworthy smartphone operating system or a TEE. Instead, the PIN to be entered is calculated by the user using a simple, invertible function on the secret PINs of his SE and RootID. Only the intended SE is then able to calculate the secret PIN necessary to establish a channel to the user's RootID and to proceed with the ID derivation. We also introduced a reverse secure combined PIN entry which can be used to provide access protection for the newly derived IDs even on a compromised smartphone.

We implemented a prototype running on a Samsung Galaxy SIII utilizing a microSD card SE. Identities can be derived from the German identity card nPA involving a Trusted Identity Provider connected through the Internet.

All in all, our approach provides a higher level of security than any pure software solution running on the application processor of a mobile device including TEE-based solutions are able to.

8. ACKNOWLEDGMENTS

This work was funded by the Bundesdruckerei GmbH.

9. REFERENCES

[1] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith. Smudge Attacks on Smartphone Touch Screens. In *WOOT'10: Proceedings of the 4th USENIX conference on Offensive technologies*. USENIX Association, Aug. 2010.

[2] BeID - Berlin electronic IDentity laboratory. eIDClientCore, July 2013. Retrieved July 19, 2013 from `http://sar.informatik.hu-berlin.de/BeID-lab/eIDClientCore/`.

[3] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. In *Research in Security and Privacy, 1992. Proceedings., 1992 IEEE Computer Society Symposium on*, pages 72–84, 1992.

[4] V. Boyko, P. MacKenzie, and S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In *Advances in Cryptology — EUROCRYPT 2000*, pages 156–171. Springer Berlin Heidelberg, May 2000.

[5] W.-D. Chen, K. E. Mayes, Y.-H. Lien, and J.-H. Chiu. NFC mobile payment with Citizen Digital Certificate. In *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on*, pages 120–126, 2011.

[6] L. Davi, A. Dmitrienko, A.-R. Sadeghi, and M. Winandy. Privilege Escalation Attacks on Android. In *ISC'10: Proceedings of the 13th international conference on Information security*. Springer Berlin Heidelberg, Oct. 2010.

[7] A. Dmitrienko, A.-R. Sadeghi, S. Tamrakar, and C. Wachsmann. SmartTokens: Delegable Access Control with NFC-Enabled Smartphones. In *Trust and Trustworthy Computing*, volume 7344 of *Lecture Notes in Computer Science*, pages 219–238. Springer Berlin Heidelberg, 2012.

[8] A. Emigh. Online Identity Theft: Phishing Technology, Chokepoints and Countermeasures. *ITTC Report on Online Identity Theft Technology and Countermeasures*, Oct. 2005.

[9] Federal Office for Information Security (BSI). BSI TR-03110, Advanced Security Mechanisms for Machine Readable Travel Documents, Mar. 2012.

[10] O. Goldreich and Y. Lindell. Session-Key Generation Using Human Passwords Only. In *Advances in Cryptology — CRYPTO 2001*, pages 408–432. Springer Berlin Heidelberg, Aug. 2001.

[11] K. Hyppönen. An Open Mobile Identity Tool: An Architecture for Mobile Identity Management. In *EuroPKI '08: Proceedings of the 5th European PKI workshop on Public Key Infrastructure: Theory and Practice*. Springer Berlin Heidelberg, June 2008.

[12] A. Leicher, A. Schmidt, and Y. Shah. Smart OpenID: A Smart Card Based OpenID Protocol. In *Information Security and Privacy Research*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 75–86. Springer Berlin Heidelberg, 2012.

[13] P. Urien, E. Marie, and C. Kiennert. A New Convergent Identity System Based on EAP-TLS Smart Cards. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, 2011.